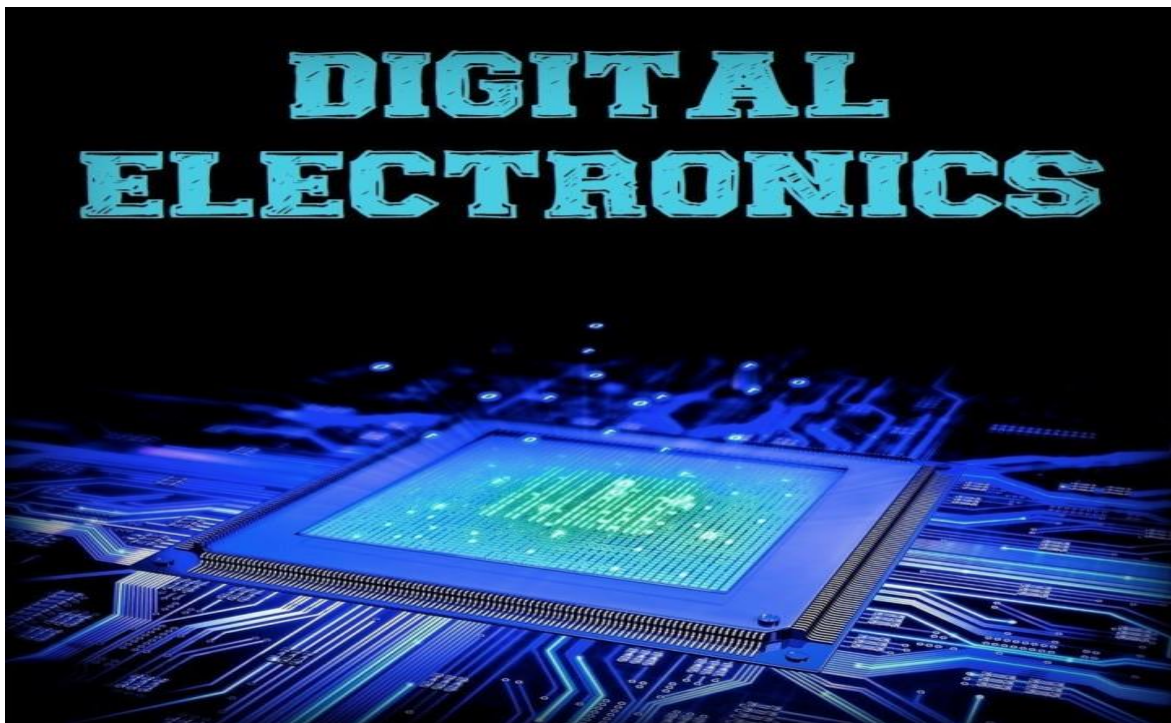


# *Abanob Evram*

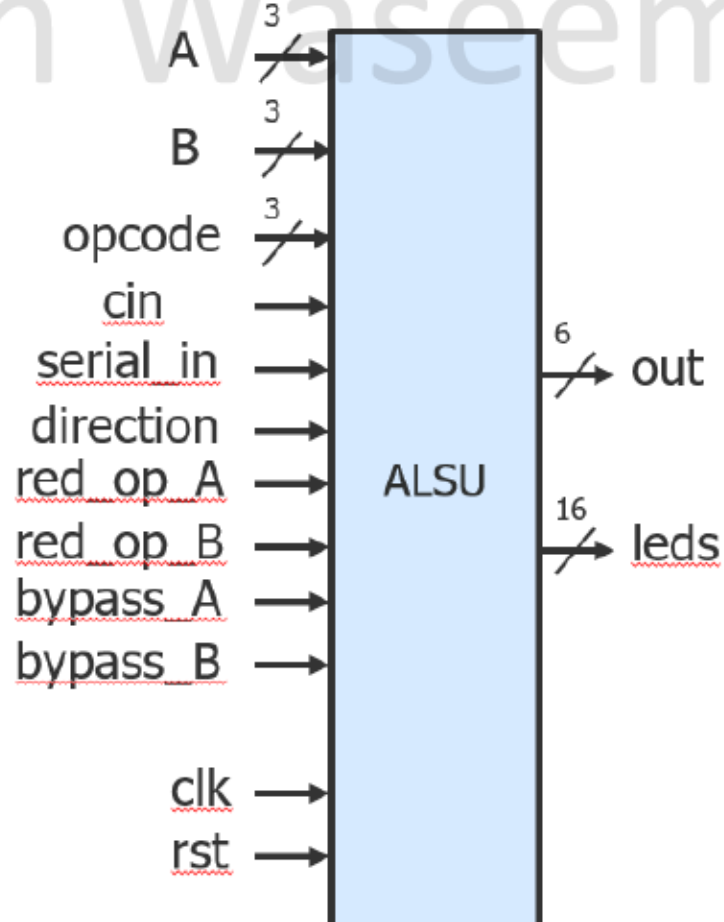
## *Assignmen4*



[Q1]

1) ALSU is a logic unit that can perform logical, arithmetic, and shift operations on input ports

- Input ports A and B have various operations that can take place depending on the value of the opcode.
- Each input bit except for the clk and rst will be sampled at the rising edge before any processing so a D-FF is expected for each input bit at the design entry.
- The output of the ALSU is registered and is available at the rising edge of the clock.



## Inputs

Each input bit except for the clk and rst will have a DFF in front of its port. Any processing will take place from the DFF output.

Input	Width	Description
clk	1	Input clock
rst	1	Active high asynchronous reset
A	3	Input port A
B	3	Input port B
cin	1	Carry in bit, only valid to be used if the parameter FULL_ADDER is "ON"
serial_in	1	Serial in bit, used in shift operations only
red_op_A	1	When set to high, this indicates that reduction operation would be executed on A rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
red_op_B	1	When set to high, this indicates that reduction operation would be executed on B rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
opcode	3	Opcode has a separate table to describe the different operations executed
bypass_A	1	When set to high, this indicates that port A will be registered to the output ignoring the opcode operation
bypass_B	1	When set to high, this indicates that port B will be registered to the output ignoring the opcode operation
direction	1	The direction of the shift or rotation operation is left when this input is set to high; otherwise, it is right.

## Outputs and parameters

Output	Width	Description
leds	16	When an invalid operation occurs, all bits blink (bits turn on and then off with each clock cycle). Blinking serves as a warning; otherwise, if a valid operation occurs, it is set to low.
out	6	Output of the ALSU

Parameter	Default value	Description
INPUT_PRIORITY	A	Priority is given to the port set by this parameter whenever there is a conflict. Conflicts can occur in two scenarios, red_op_A and red_op_B are both set to high or bypass_A and bypass_B are both set to high. Legal values for this parameter are A and B
FULL_ADDER	ON	When this parameter has value "ON" then cin input must be considered in the addition operation between A and B. Legal values for this parameter are ON and OFF

## Opcodes & Handling invalid cases

### Invalid cases

1. Opcode bits are set to 110 or 111
2. red\_op\_A or red\_op\_B are set to high and the opcode is not AND or XOR operation

### Output when invalid cases occurs

1. leds are blinking
2. out bits are set to low, but if the bypass\_A or bypass\_B are high then the output will take the value of A or B.

Opcode	Operation
000	AND
001	XOR
010	Addition
011	Multiplication
100	Shift output by 1 bit
101	Rotate output by 1 bit
110	Invalid opcode
111	Invalid opcode

## The design code:

```
1 module ALSU(A,B,opcode,cin,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst,out,leds);
2   parameter INPUT_PRIORITY="A";//A OR B
3   parameter FULL_ADDER="ON";//ON OR OFF
4   input cin,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst;
5   input [2:0] A,B,opcode;
6   output reg [5:0] out ;
7   output reg [15:0] leds;
8   reg [2:0] Q_A,Q_B,Q_opcode;
9   reg Q_cin,Q_serial_in,Q_direction,Q_red_op_A,Q_red_op_B,Q_bypass_A,Q_bypass_B;
10
11   always @(posedge clk or posedge rst) begin
12       if (rst) begin
13           Q_A<=0;
14           Q_B<=0;
15           Q_opcode<=0;
16           Q_cin<=0;
17           Q_serial_in<=0;
18           Q_direction<=0;
19           Q_red_op_A<=0;
20           Q_red_op_B<=0;
21           Q_bypass_A<=0;
22           Q_bypass_B<=0;
23       end
24       else begin
25           Q_A<=A;
26           Q_B<=B;
27           Q_opcode<=opcode;
28           Q_cin<=cin;
29           Q_serial_in<=serial_in;
30           Q_direction<=direction;
31           Q_red_op_A<=red_op_A;
32           Q_red_op_B<=red_op_B;
33           Q_bypass_A<=bypass_A;
34           Q_bypass_B<=bypass_B;
35       end
36   end
37
38   always @(posedge clk or posedge rst) begin
39       if(rst)
40           out<=0;
41
42       else begin
43           if(Q_bypass_A&&Q_bypass_B) begin
44               if(INPUT_PRIORITY=="B")
45                   out<=Q_B;
46               else //default : INPUT_PRIORITY = A
47                   out<=Q_A;
48           end
49           else if(Q_bypass_A)
50               out<=Q_A;
51           else if(Q_bypass_B)
52               out<=Q_B;
53
54           else begin
55               case(Q_opcode)
```

```

55 ▼      case(Q_opcode)
56 ▼          0: begin //Operation: AND
57 ▼              if (Q_red_op_A&&Q_red_op_B) begin
58 ▼                  if (INPUT_PRIORITY=="B")
59 ▼                      out<=&Q_B;
60 ▼                  else
61 ▼                      out<=&Q_A;
62 ▼              end
63 ▼          else if (Q_red_op_A)
64 ▼              out<=&Q_A;
65 ▼          else if (Q_red_op_B)
66 ▼              out<=&Q_B;
67 ▼          else
68 ▼              out<=Q_A&Q_B;
69 ▼      end
70 ▼      1: begin //Operation: XOR
71 ▼          if (Q_red_op_A&&Q_red_op_B) begin
72 ▼              if (INPUT_PRIORITY=="B")
73 ▼                  out<=~Q_B;
74 ▼              else
75 ▼                  out<=~Q_A;
76 ▼          end
77 ▼          else if (Q_red_op_A)
78 ▼              out<=~Q_A;
79 ▼          else if (Q_red_op_B)
80 ▼              out<=~Q_B;
81 ▼          else
82 ▼              out<=Q_A^Q_B;
83 ▼      end
84 ▼      2: begin //Operation: Addition
85 ▼          if(Q_red_op_A||Q_red_op_B)
86 ▼              out<=0;
87 ▼          else if (FULL_ADDER=="OFF")
88 ▼              out<=Q_A+Q_B;
89 ▼          else //default : FULL_ADDER= ON
90 ▼              out<=Q_A+Q_B+Q_cin;
91 ▼      end
92 ▼      3: begin //Operation: Multiplication
93 ▼          if(Q_red_op_A||Q_red_op_B)
94 ▼              out<=0;
95 ▼          else
96 ▼              out<=Q_A*Q_B;
97 ▼      end
98 ▼      4: begin //Operation: Shift output by 1 bit
99 ▼          if(Q_red_op_A||Q_red_op_B)
100 ▼              out<=0;
101 ▼          else begin
102 ▼              if (Q_direction) // if direction ==1 then shift left
103 ▼                  out<={out[4:0],Q_serial_in};
104 ▼              else //shift right
105 ▼                  out<={Q_serial_in,out[5:1]};
106 ▼              end
107 ▼          end
108 ▼      5: begin //Operation: Rotate output by 1 bit
109 ▼          if(Q_red_op_A||Q_red_op_B)

```

```

85         if(Q_red_op_A||Q_red_op_B)
86             out<=0;
87         else if (FULL_ADDER=="OFF")
88             out<=Q_A+Q_B;
89         else //default : FULL_ADDER= ON
90             out<=Q_A+Q_B+Q_cin;
91         end
92         3: begin //Operation: Multiplication
93             if(Q_red_op_A||Q_red_op_B)
94                 out<=0;
95             else
96                 out<=Q_A*Q_B;
97             end
98         4: begin //Operation: Shift output by 1 bit
99             if(Q_red_op_A||Q_red_op_B)
100                 out<=0;
101             else begin
102                 if (Q_direction) // if direction ==1 then shift left
103                     out<={out[4:0],Q_serial_in};
104                 else //shift right
105                     out<={Q_serial_in,out[5:1]};
106                 end
107             end
108         5: begin //Operation: Rotate output by 1 bit
109             if(Q_red_op_A||Q_red_op_B)
110                 out<=0;
111             else begin
112                 if (Q_direction) // if direction ==1 then Rotate left
113                     out<={out[4:0],out[5]};
114                 else //Rotate right
115                     out<={out[0],out[5:1]};
116                 end
117             end
118             default : out<=0;
119         endcase
120     end
121 end
122
123 always @(posedge clk or posedge rst) begin
124     if (rst)
125         leds<=0;
126     else if (Q_opcode==0||Q_opcode==1)
127         leds<=0;
128     else if (Q_opcode==2||Q_opcode==3||Q_opcode==4||Q_opcode==5) begin
129         if(Q_red_op_A||Q_red_op_B) begin
130             leds<=~leds;
131         end
132     else
133         leds<=0;
134     end
135     else begin
136         leds<=~leds;
137     end
138 end
139 endmodule

```

## The testbench code:

```
1 module ALSU_tb();
2     parameter INPUT_PRIORITY_TB="A";//A OR B
3     parameter FULL_ADDER_TB="ON";//ON OR OFF
4     reg cin_tb,serial_in_tb,direction_tb,red_op_A_tb,red_op_B_tb,bypass_A_tb,bypass_B_tb,clk_tb,rst_tb;
5     reg [2:0] A_tb,B_tb,opcode_tb;
6     wire [5:0] out_dut;
7     wire [15:0] leds_dut;
8
9     ALSU #(INPUT_PRIORITY_TB,FULL_ADDER_TB) dut(A_tb,B_tb,opcode_tb,cin_tb,serial_in_tb,direction_tb,red_op_A_tb,red_op_B_tb,bypass_A_tb,bypass_B_tb,clk_tb,rst_tb,out_dut,leds_dut);
10    integer i;
11    initial begin
12        clk_tb=0;
13        forever
14            #1 clk_tb=~clk_tb;
15    end
16    initial begin
17        rst_tb=1;cin_tb=0;serial_in_tb=0;direction_tb=0;red_op_A_tb=0;red_op_B_tb=0;bypass_A_tb=0;bypass_B_tb=0; // test if rst = 1
18        for(i=0;i<30;i=i+1) begin
19            A_tb=$random;
20            B_tb=$random;
21            opcode_tb=$random;
22            @(negedge clk_tb);
23        end
24        rst_tb=0;bypass_A_tb=1;bypass_B_tb=1; //test if bypass A&B =1
25        for(i=0;i<30;i=i+1) begin
26            A_tb=$random;
27            B_tb=$random;
28            opcode_tb=$random;
29            @(negedge clk_tb);
30        end
31        bypass_B_tb=0; //test if bypass A=1
32        for(i=0;i<30;i=i+1) begin
33            A_tb=$random;
34            B_tb=$random;
35            opcode_tb=$random;
36            @(negedge clk_tb);
37        end
38        bypass_B_tb=1;bypass_A_tb=0; //test if bypass B=1
39        for(i=0;i<30;i=i+1) begin
40            A_tb=$random;
41            B_tb=$random;
42            opcode_tb=$random;
43            @(negedge clk_tb);
44        end
45
46        bypass_B_tb=0;red_op_A_tb=1;red_op_B_tb=1;opcode_tb=0; //test AND if red_op_A&B=1 :(INPUT_PRIORITY=A)
47        for(i=0;i<30;i=i+1) begin
48            A_tb=$random;
49            B_tb=$random;
50            @(negedge clk_tb);
51        end
52        red_op_A_tb=0; //test AND if red_op_B=1
53        for(i=0;i<30;i=i+1) begin
54            A_tb=$random;
55            B_tb=$random;
```

```

52     red_op_A_tb=0;//test AND if red_op_B=1
53     for(i=0;i<30;i=i+1) begin
54         A_tb=$random;
55         B_tb=$random;
56         @(negedge clk_tb);
57     end
58     red_op_B_tb=0;//test AND if red_op_B=0&red_op_B=0
59     for(i=0;i<30;i=i+1) begin
60         A_tb=$random;
61         B_tb=$random;
62         @(negedge clk_tb);
63     end
64
65     red_op_A_tb=1;red_op_B_tb=1;opcode_tb=1; //test XOR if red_op_A&B=1 :(INPUT_PRIORITY=A)
66     for(i=0;i<30;i=i+1) begin
67         A_tb=$random;
68         B_tb=$random;
69         @(negedge clk_tb);
70     end
71     red_op_A_tb=0;//test XOR if red_op_B=1
72     for(i=0;i<30;i=i+1) begin
73         A_tb=$random;
74         B_tb=$random;
75         @(negedge clk_tb);
76     end
77     red_op_B_tb=0;//test XOR if red_op_B=0&red_op_B=0
78     for(i=0;i<30;i=i+1) begin
79         A_tb=$random;
80         B_tb=$random;
81         @(negedge clk_tb);
82     end
83
84     opcode_tb=2; //test addition
85     for(i=0;i<30;i=i+1) begin
86         A_tb=$random;
87         B_tb=$random;
88         cin_tb=$random;
89         @(negedge clk_tb);
90     end
91
92     opcode_tb=3; //test multiplication
93     for(i=0;i<30;i=i+1) begin
94         A_tb=$random;
95         B_tb=$random;
96         @(negedge clk_tb);
97     end
98
99     opcode_tb=4;direction_tb=1;//test shift left
100    for(i=0;i<30;i=i+1) begin
101        serial_in_tb=$random;
102        @(negedge clk_tb);
103    end
104    direction_tb=0; //test shift right
105    for(i=0;i<30;i=i+1) begin
106        serial_in_tb=$random;
107        @(negedge clk_tb);

```



```

76         end
77         red_op_B_tb=0;//test XOR if red_op_B=0&red_op_B=0
78         for(i=0;i<30;i=i+1) begin
79             A_tb=$random;
80             B_tb=$random;
81             @(negedge clk_tb);
82         end
83
84         opcode_tb=2; //test addition
85         for(i=0;i<30;i=i+1) begin
86             A_tb=$random;
87             B_tb=$random;
88             cin_tb=$random;
89             @(negedge clk_tb);
90         end
91
92         opcode_tb=3; //test multiplication
93         for(i=0;i<30;i=i+1) begin
94             A_tb=$random;
95             B_tb=$random;
96             @(negedge clk_tb);
97         end
98
99         opcode_tb=4;direction_tb=1;//test shift left
100        for(i=0;i<30;i=i+1) begin
101            serial_in_tb=$random;
102            @(negedge clk_tb);
103        end
104        direction_tb=0; //test shift right
105        for(i=0;i<30;i=i+1) begin
106            serial_in_tb=$random;
107            @(negedge clk_tb);
108        end
109
110        opcode_tb=5;//test rotate right
111        for(i=0;i<30;i=i+1) begin
112            serial_in_tb=$random;
113            @(negedge clk_tb);
114        end
115        direction_tb=1;//test rotate left
116        for(i=0;i<30;i=i+1) begin
117            serial_in_tb=$random;
118            @(negedge clk_tb);
119        end
120        red_op_A_tb=1;red_op_B_tb=1;// test some invalid cases
121        for(i=0;i<100;i=i+1) begin
122            A_tb=$random;
123            B_tb=$random;
124            opcode_tb=$urandom_range(2,7);
125            @(negedge clk_tb);
126        end
127        $stop;
128    end
129 endmodule

```

## The do file code:

```
1 |vlib work
2
3 |vlog ALSU.v ALSU_tb.v
4
5 |vsim -voptargs=+acc ALSU_tb
6
7 |add wave *
8
9 |run -all
10
11 |#quit -sim
```

## The constraint code:

```
6 |## Clock signal
7 |set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports clk]
8 |create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]
9
10
11 |## Switches
12 |set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {opcode[0]}]
13 |set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {opcode[1]}]
14 |set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMOS33} [get_ports {opcode[2]}]
15 |set_property -dict {PACKAGE_PIN W17 IOSTANDARD LVCMOS33} [get_ports {A[0]}]
16 |set_property -dict {PACKAGE_PIN W15 IOSTANDARD LVCMOS33} [get_ports {A[1]}]
17 |set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {A[2]}]
18 |set_property -dict {PACKAGE_PIN W14 IOSTANDARD LVCMOS33} [get_ports {B[0]}]
19 |set_property -dict {PACKAGE_PIN W13 IOSTANDARD LVCMOS33} [get_ports {B[1]}]
20 |set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {B[2]}]
21 |set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports cin]
22 |set_property -dict {PACKAGE_PIN T2 IOSTANDARD LVCMOS33} [get_ports red_op_A]
23 |set_property -dict {PACKAGE_PIN R3 IOSTANDARD LVCMOS33} [get_ports red_op_B]
24 |set_property -dict {PACKAGE_PIN W2 IOSTANDARD LVCMOS33} [get_ports bypass_A]
25 |set_property -dict {PACKAGE_PIN U1 IOSTANDARD LVCMOS33} [get_ports bypass_B]
26 |set_property -dict {PACKAGE_PIN T1 IOSTANDARD LVCMOS33} [get_ports direction]
27 |set_property -dict {PACKAGE_PIN R2 IOSTANDARD LVCMOS33} [get_ports serial_in]
28
29
30 |## LEDs
31 |set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {leds[0]}]
32 |set_property -dict {PACKAGE_PIN E19 IOSTANDARD LVCMOS33} [get_ports {leds[1]}]
33 |set_property -dict {PACKAGE_PIN U19 IOSTANDARD LVCMOS33} [get_ports {leds[2]}]
34 |set_property -dict {PACKAGE_PIN V19 IOSTANDARD LVCMOS33} [get_ports {leds[3]}]
35 |set_property -dict {PACKAGE_PIN W18 IOSTANDARD LVCMOS33} [get_ports {leds[4]}]
36 |set_property -dict {PACKAGE_PIN U15 IOSTANDARD LVCMOS33} [get_ports {leds[5]}]
37 |set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {leds[6]}]
38 |set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {leds[7]}]
39 |set_property -dict {PACKAGE_PIN V13 IOSTANDARD LVCMOS33} [get_ports {leds[8]}]
40 |set_property -dict {PACKAGE_PIN V3 IOSTANDARD LVCMOS33} [get_ports {leds[9]}]
41 |set_property -dict {PACKAGE_PIN W3 IOSTANDARD LVCMOS33} [get_ports {leds[10]}]
42 |set_property -dict {PACKAGE_PIN U3 IOSTANDARD LVCMOS33} [get_ports {leds[11]}]
43 |set_property -dict {PACKAGE_PIN P3 IOSTANDARD LVCMOS33} [get_ports {leds[12]}]
44 |set_property -dict {PACKAGE_PIN N3 IOSTANDARD LVCMOS33} [get_ports {leds[13]}]
45 |set_property -dict {PACKAGE_PIN P1 IOSTANDARD LVCMOS33} [get_ports {leds[14]}]
46 |set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {leds[15]}]
47
```

```

152 ## Configuration options, can be used for all designs
153 set_property CONFIG_VOLTAGE 3.3 [current_design]
154 set_property CFGBVS VCCO [current_design]
155
156 ## SPI configuration mode options for QSPI boot, can be used for all designs
157 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
158 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
159 set_property CONFIG_MODE SPIx4 [current_design]
160
161 create_debug_core u_ila_0 ila
162 set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
163 set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
164 set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
165 set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
166 set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
167 set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
168 set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
169 set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
170 set_property port_width 1 [get_debug_ports u_ila_0/clock]
171 connect_debug_port u_ila_0/clock [get_nets [list clk_IBUF_IBUF]]
172 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
173 set_property port_width 3 [get_debug_ports u_ila_0/probe0]
174 connect_debug_port u_ila_0/probe0 [get_nets [list {B_IBUF[0]} {B_IBUF[1]} {B_IBUF[2]}]]
175 create_debug_port u_ila_0 probe
176 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
177 set_property port_width 16 [get_debug_ports u_ila_0/probe1]
178 connect_debug_port u_ila_0/probe1 [get_nets [list {leds_OBUF[0]} {leds_OBUF[1]} {leds_OBUF[2]} {leds_OBUF[3]} {leds_OBUF[4]} {leds_OBUF[5]} {leds_OBUF[6]}
179 {leds_OBUF[7]} {leds_OBUF[8]} {leds_OBUF[9]} {leds_OBUF[10]} {leds_OBUF[11]} {leds_OBUF[12]} {leds_OBUF[13]} {leds_OBUF[14]} {leds_OBUF[15]}]]
180 create_debug_port u_ila_0 probe
181 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
182 set_property port_width 3 [get_debug_ports u_ila_0/probe2]
183 connect_debug_port u_ila_0/probe2 [get_nets [list {opcode_IBUF[0]} {opcode_IBUF[1]} {opcode_IBUF[2]}]]
184 create_debug_port u_ila_0 probe
185 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
186 set_property port_width 6 [get_debug_ports u_ila_0/probe3]
187 connect_debug_port u_ila_0/probe3 [get_nets [list {out_OBUF[0]} {out_OBUF[1]} {out_OBUF[2]} {out_OBUF[3]} {out_OBUF[4]} {out_OBUF[5]}]]
188 create_debug_port u_ila_0 probe
189 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
190 set_property port_width 3 [get_debug_ports u_ila_0/probe4]
191 connect_debug_port u_ila_0/probe4 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]}]]
192 create_debug_port u_ila_0 probe
193 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
194 set_property port_width 1 [get_debug_ports u_ila_0/probe5]
195 connect_debug_port u_ila_0/probe5 [get_nets [list bypass_A_IBUF]]
196 create_debug_port u_ila_0 probe
197 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
198 set_property port_width 1 [get_debug_ports u_ila_0/probe6]
199 connect_debug_port u_ila_0/probe6 [get_nets [list bypass_B_IBUF]]

```

```

185 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
186 set_property port_width 6 [get_debug_ports u_ila_0/probe3]
187 connect_debug_port u_ila_0/probe3 [get_nets [list {out_OBUF[0]} {out_OBUF[1]} {out_OBUF[2]} {out_OBUF[3]} {out_OBUF[4]} {out_OBUF[5]}]]
188 create_debug_port u_ila_0 probe
189 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
190 set_property port_width 3 [get_debug_ports u_ila_0/probe4]
191 connect_debug_port u_ila_0/probe4 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]}]]
192 create_debug_port u_ila_0 probe
193 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
194 set_property port_width 1 [get_debug_ports u_ila_0/probe5]
195 connect_debug_port u_ila_0/probe5 [get_nets [list bypass_A_IBUF]]
196 create_debug_port u_ila_0 probe
197 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
198 set_property port_width 1 [get_debug_ports u_ila_0/probe6]
199 connect_debug_port u_ila_0/probe6 [get_nets [list bypass_B_IBUF]]
200 create_debug_port u_ila_0 probe
201 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe7]
202 set_property port_width 1 [get_debug_ports u_ila_0/probe7]
203 connect_debug_port u_ila_0/probe7 [get_nets [list cin_IBUF]]
204 create_debug_port u_ila_0 probe
205 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe8]
206 set_property port_width 1 [get_debug_ports u_ila_0/probe8]
207 connect_debug_port u_ila_0/probe8 [get_nets [list clk_IBUF]]
208 create_debug_port u_ila_0 probe
209 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe9]
210 set_property port_width 1 [get_debug_ports u_ila_0/probe9]
211 connect_debug_port u_ila_0/probe9 [get_nets [list direction_IBUF]]
212 create_debug_port u_ila_0 probe
213 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe10]
214 set_property port_width 1 [get_debug_ports u_ila_0/probe10]
215 connect_debug_port u_ila_0/probe10 [get_nets [list red_op_A_IBUF]]
216 create_debug_port u_ila_0 probe
217 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe11]
218 set_property port_width 1 [get_debug_ports u_ila_0/probe11]
219 connect_debug_port u_ila_0/probe11 [get_nets [list red_op_B_IBUF]]
220 create_debug_port u_ila_0 probe
221 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe12]
222 set_property port_width 1 [get_debug_ports u_ila_0/probe12]
223 connect_debug_port u_ila_0/probe12 [get_nets [list rst_IBUF]]
224 create_debug_port u_ila_0 probe
225 set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe13]
226 set_property port_width 1 [get_debug_ports u_ila_0/probe13]
227 connect_debug_port u_ila_0/probe13 [get_nets [list serial_in_IBUF]]
228 set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
229 set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
230 set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
231 connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
232

```

## ##Buttons

```

set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports rst]
#set_property -dict { PACKAGE_PIN T18      IOSTANDARD LVCMOS33 } [get_ports btnU]
#set_property -dict { PACKAGE_PIN W19      IOSTANDARD LVCMOS33 } [get_ports btnL]
#set_property -dict { PACKAGE_PIN T17      IOSTANDARD LVCMOS33 } [get_ports btnR]
#set_property -dict { PACKAGE_PIN U17      IOSTANDARD LVCMOS33 } [get_ports btnD]

```

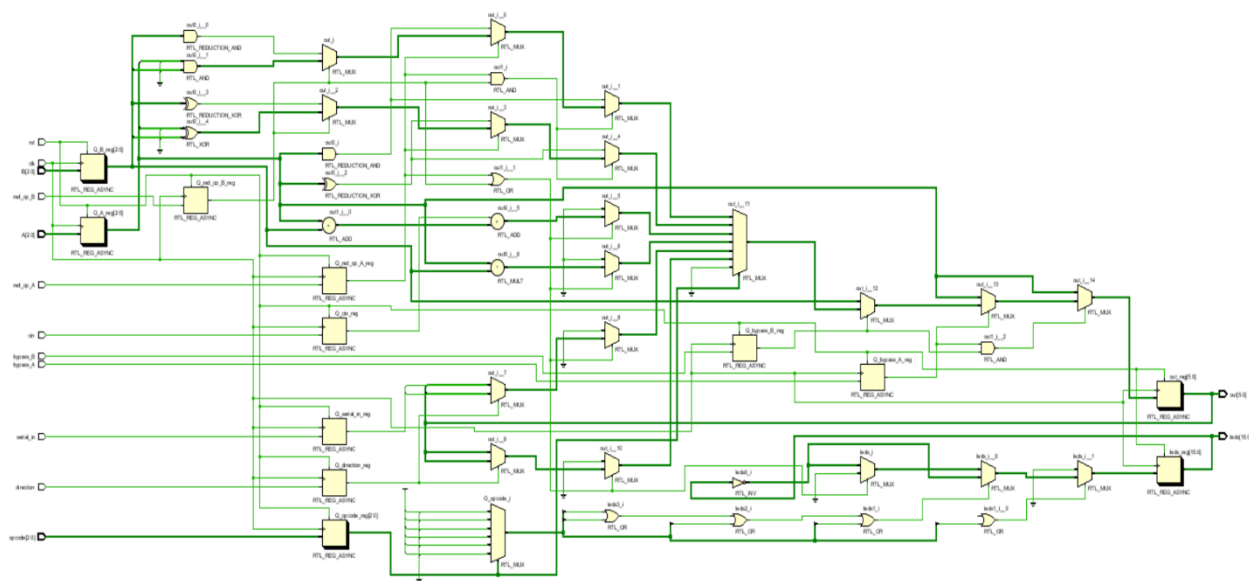
## 1-Elaboration (messages\_tab)

Tcl Console Messages Log Reports Design Runs

☒ Info (8)
 ☐ Status (11)
 Show All

- Vivado Commands (3 infos)
  - General Messages (3 infos)
    - [IP\_Flow 19-234] Refreshing IP repositories
    - [IP\_Flow 19-1704] No user IP repositories specified
    - [IP\_Flow 19-2313] Loaded Vivado IP repository 'D:/Vivado/Vivado/2018.2/data/ip'.
- Elaborated Design (5 infos)
  - General Messages (5 infos)
    - [Synth 8-6157] synthesizing module 'ALSU' [ALSU.v:1]
    - [Synth 8-6155] done synthesizing module 'ALSU' (1#1) [ALSU.v:1]
    - [Project 1-570] Preparing netlist for logic optimization
    - [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
    - [Project 1-111] Unisim Transformation Summary:  
No Unisim elements were transformed.

## 2-Elaboration(Schematic)



### 3-synthesis(messages\_tab)

The screenshot shows the 'Messages' tab in the synthesis tool. The top bar includes 'Tcl Console', 'Messages', 'Log', 'Reports', and 'Design Runs'. Below the bar, there are search and filter icons, and a summary of message counts: 'Warning (1)', 'Info (25)', and 'Status (24)'. A 'Show All' button is on the right. The main area displays a warning message under the 'Synthesis (1 warning)' category: '[Constraints 18-5210] No constraint will be written out.'

### 4-synthesis(Utilization\_report)

The screenshot shows the 'Utilization' report in the synthesis tool. The top bar includes 'Tcl Console', 'Messages', 'Log', 'Reports', 'Design Runs', 'Utilization', and 'Debug'. The left sidebar shows a hierarchy of utilization categories: 'Hierarchy', 'Summary', 'Slice Logic', 'Slice LUTs (<1%)', 'LUT as Logic (<1%)', 'Slice Registers (<1%)', 'Register as Flip Flop (', 'Memory', 'DSP', 'IO and GT Specific', 'Bonded IOB (38%)', 'IOB Master Pads', and 'IOB Slave Pads'. The main area displays a table with the following data:

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
ALSU	55	38	40	1

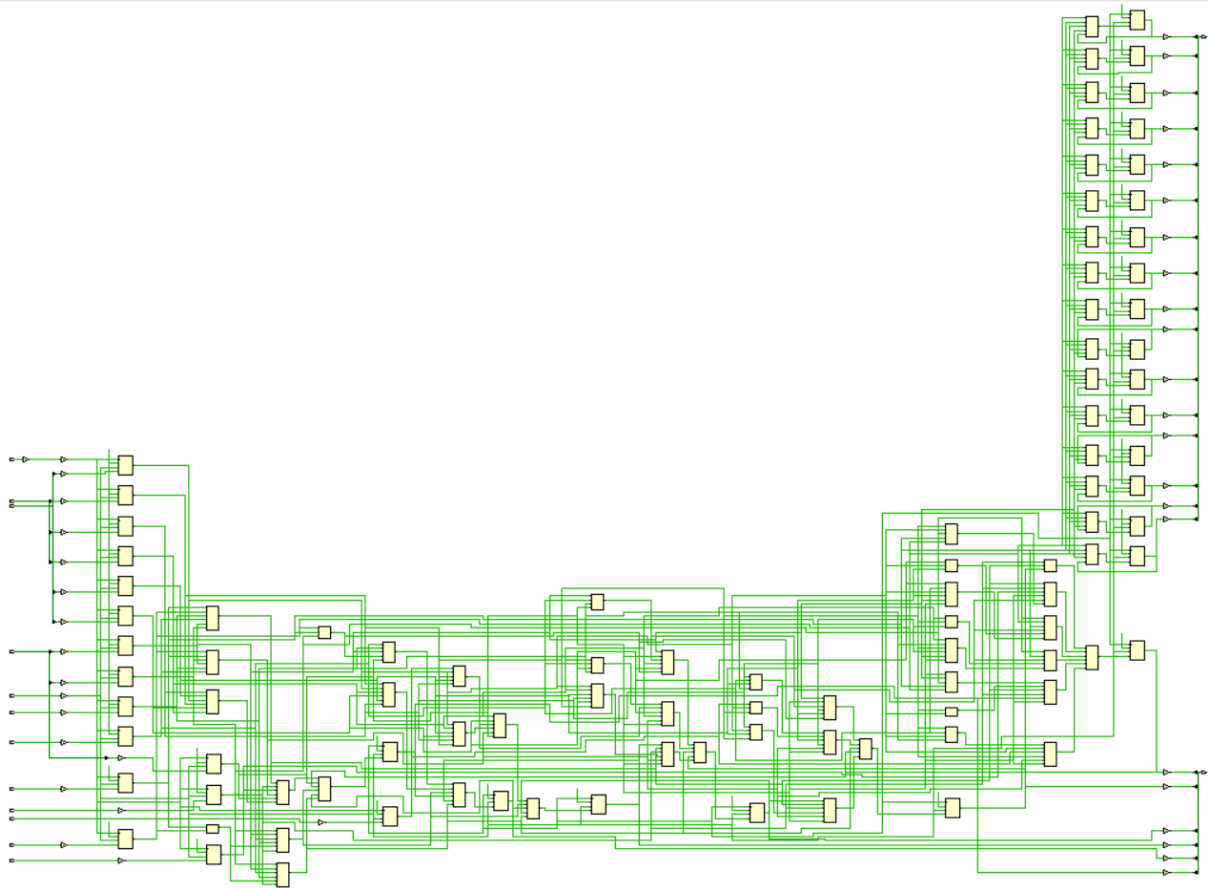
### 5-synthesis(Timing\_report)

The screenshot shows the 'Timing' report in the synthesis tool. The top bar includes 'Tcl Console', 'Messages', 'Log', 'Reports', 'Design Runs', 'Timing', 'Utilization', and 'Debug'. The left sidebar shows a hierarchy of timing categories: 'General Information', 'Timer Settings', 'Design Timing Summary', 'Clock Summary (1)', 'Check Timing (39)', 'Intra-Clock Paths', 'Inter-Clock Paths', 'Other Path Groups', 'User Ignored Paths', and 'Unconstrained Paths'. The main area displays a 'Design Timing Summary' table with the following data:

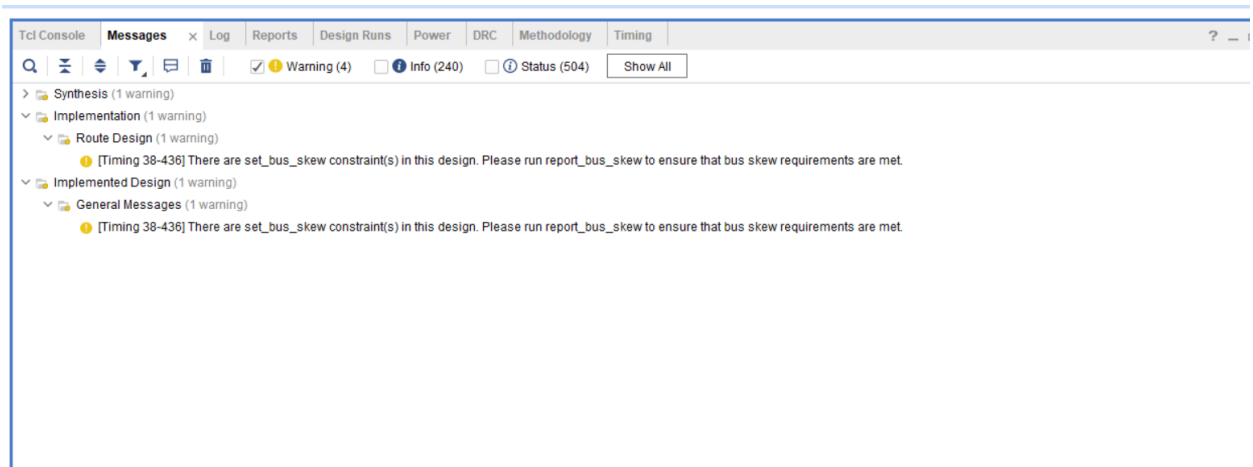
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.776 ns	Worst Hold Slack (WHS): 0.139 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 22	Total Number of Endpoints: 22	Total Number of Endpoints: 39

All user specified timing constraints are met.

## 6-synthesis(Schematic)



## 7-implmentation(messages\_tab)





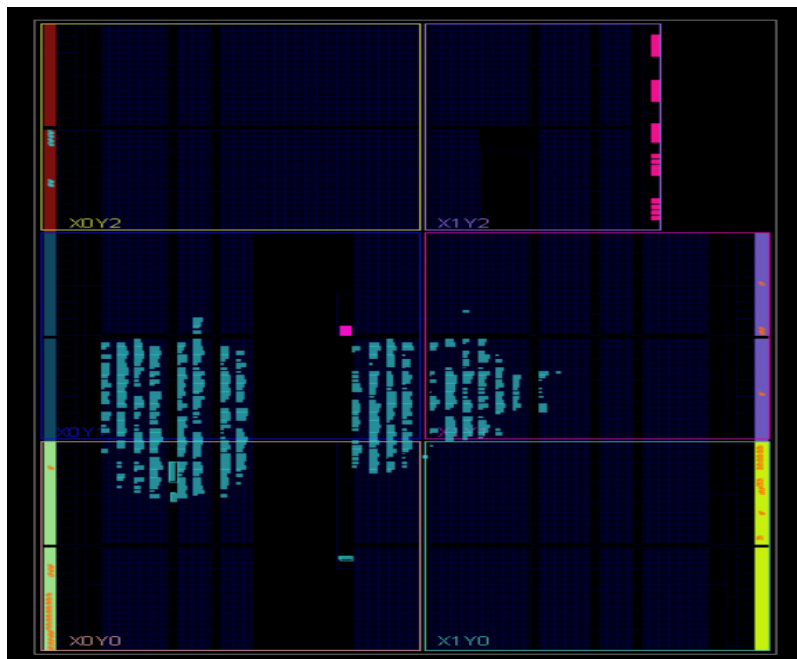
## 8-Implementation(Utilization\_report)

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (815 0)	LUT as Logic (20800)	LUT as Memory (9600)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUF/GCTRL (32)	BSCAN/E2 (4)
ALSU	1704	2545	23	839	1527	177	1059	1.5	40	2	1
dbg_hub (dbg_hub)	475	727	0	241	451	24	309	0	0	1	1
u_ila_0 (u_ila_0)	1174	1780	23	589	1021	153	727	1.5	0	0	0

## 9-Implementation(Timing\_report)

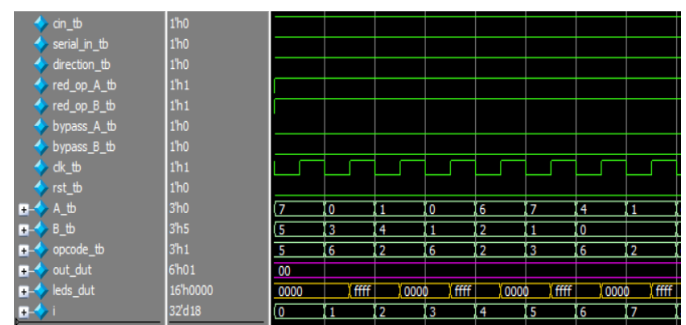
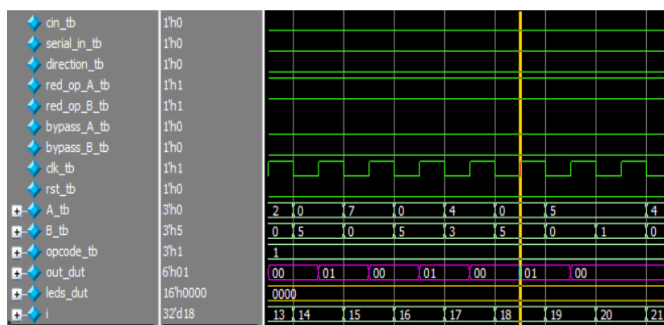
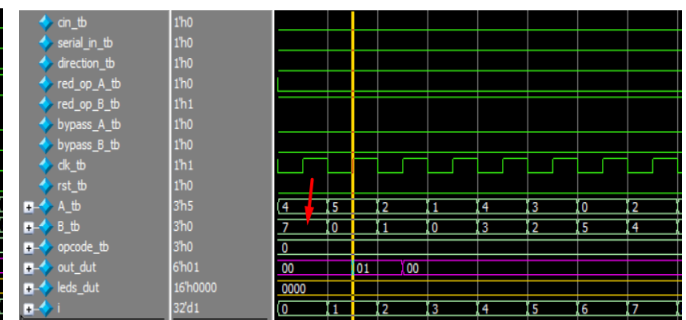
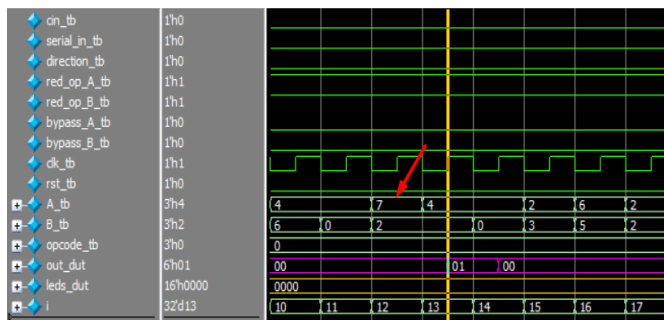
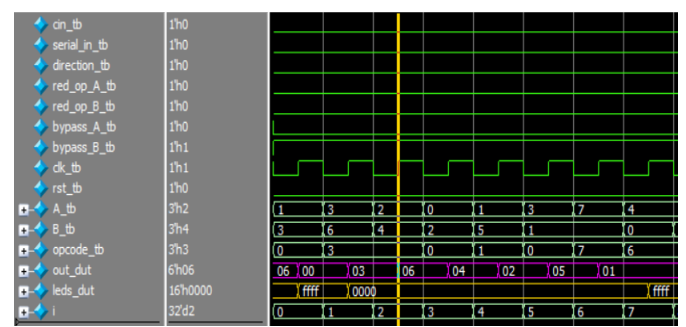
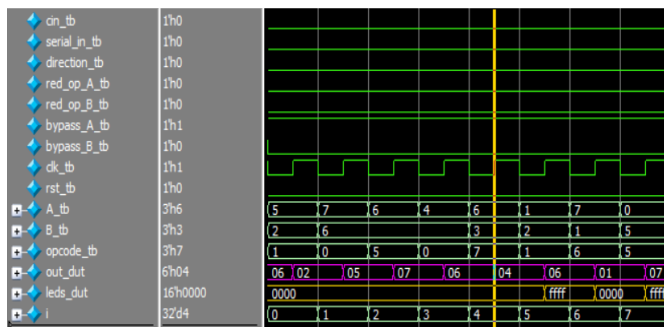
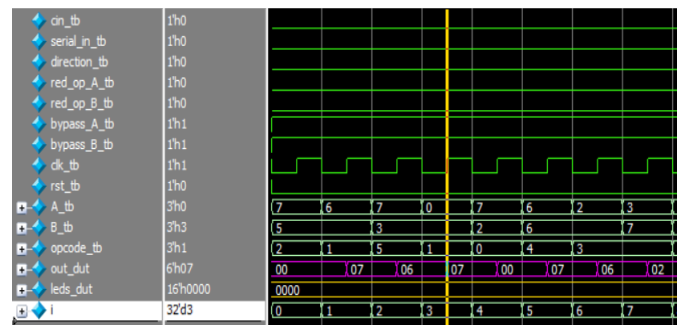
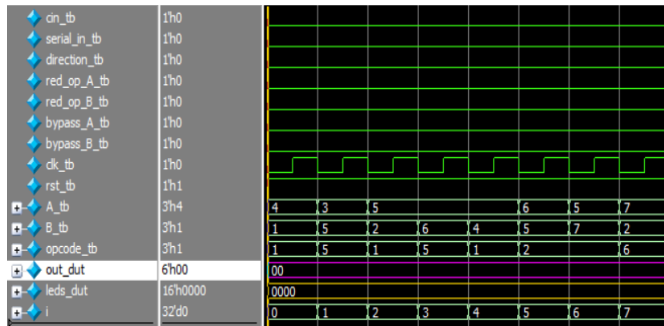
Design Timing Summary			
Setup	Hold	Pulse Width	
Worst Negative Slack (WNS): 1.318 ns	Worst Hold Slack (WHS): 0.035 ns	Worst Pulse Width Slack (WPWS): 3.750 ns	
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns	
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	
Total Number of Endpoints: 4884	Total Number of Endpoints: 4868	Total Number of Endpoints: 2864	
All user specified timing constraints are met.			

## 10-Implementation(device)



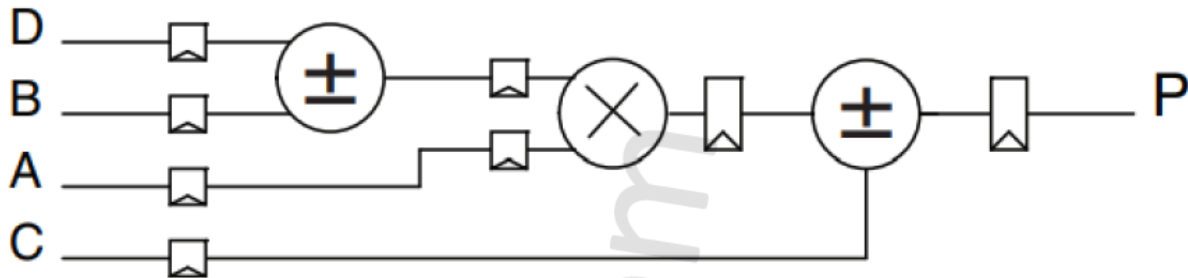


## Some snippets of waves:



## [Q2]

2) Design the following simplified version of the DSP block DSP48A1 from Xilinx Spartan-6 FPGA. Use a directed testbench to simplify the verification. Check the waveform to make sure that the operations are done correctly at every pipeline stage.



Port	Type	Width
A	Input	18
B	Input	18
C	Input	48
D	Input	18
clk	Input	1
rst_n (sync active low)	Input	1
P	Output	48

Parameters:

- OPERATION: take 2 values either "ADD" or "SUBTRACT", Default value "ADD"
  - When subtracting use "D - B" and "multiplier\_out - C". multiplier\_out is an internal signal

## The design (first code):

```
1 module Simple_DSP(A,B,C,D,clk,rst_n,P);
2   parameter OPERATION = "ADD"; //ADD OR SUBTRACT
3   input [17:0] A,B,D;
4   input [47:0] C;
5   input clk,rst_n;
6   output [47:0] P;
7   reg [17:0] Adder1 ;
8   reg [47:0] Adder2 ;
9   reg [47:0] mult;
10  wire [17:0] Q_D,Q_B,Q_A1,Q_A2,Q_Adder1;
11  wire [47:0] Q_C,Q_mult;
12
13  D_Flipflop    #(18)    d0(D,rst_n,clk,Q_D);
14  D_Flipflop    #(18)    d1(B,rst_n,clk,Q_B);
15  D_Flipflop    #(18)    d2(A,rst_n,clk,Q_A1);
16  D_Flipflop    #(48)    d3(C,rst_n,clk,Q_C);
17
18  D_Flipflop    #(18)    d4(Adder1,rst_n,clk,Q_Adder1);
19  D_Flipflop    #(18)    d5(Q_A1,rst_n,clk,Q_A2);
20  D_Flipflop    #(48)    d6(mult,rst_n,clk,Q_mult);
21  D_Flipflop    #(48)    d7(Adder2,rst_n,clk,P);
22
23  always @(*) begin
24    if (OPERATION=="SUBTRACT")
25      Adder1= Q_D - Q_B;
26
27    else
28      Adder1= Q_D + Q_B;
29
30    mult = Q_A2*Q_Adder1;
31
32    if (OPERATION=="SUBTRACT")
33      Adder2= Q_mult - Q_C;
34
35    else
36      Adder2= Q_mult + Q_C;
37  end
38 endmodule
```

```
1 module D_Flipflop(d,rstn,clk,q);
2   parameter WIDTH=1;
3   input [WIDTH-1:0] d;
4   input rstn,clk;
5   output reg [WIDTH-1:0] q;
6   always @(posedge clk) begin
7     if (~rstn)
8       q<=0;
9     else
10      q<=d;
11   end
12 endmodule
```

## The design (second code):

```
1 module Simple_DSP2(A,B,C,D,clk,rst_n,p);
2   parameter OPERATION = "ADD"; //ADD OR SUBTRACT
3   input [17:0] A,B,D;
4   input [47:0] C;
5   input clk,rst_n;
6   output reg [47:0] p;
7   reg [17:0] Q_D,Q_B,Q_A1,Q_A2,Q_Adder1;
8   reg [47:0] Q_C,Q_mult;
9   always @(posedge clk ) begin
10     if (~rst_n) begin
11       Q_D<=0;
12       Q_B<=0;
13       Q_A1<=0;
14       Q_A2<=0;
15       Q_Adder1<=0;
16       Q_C<=0;
17       Q_mult<=0;
18       p<=0;
19     end
20     else begin
21       Q_D<=D;
22       Q_B<=B;
23       Q_C<=C;
24       Q_A1<=A;
25       Q_A2<=Q_A1;
26       Q_mult<=Q_A2*Q_Adder1;
27       if(OPERATION=="SUBTRACT") begin
28         Q_Adder1<= Q_D - Q_B;
29         p<= Q_mult - Q_C;
30       end
31       else begin
32         Q_Adder1<= Q_D + Q_B;
33         p<= Q_mult + Q_C;
34       end
35     end
36   end
37 endmodule
```

## The testbench code:

```
1  module DSP_block_tb ();
2      parameter OPERATION_tb="ADD";
3      reg [17:0] D_tb,B_tb,A_tb;
4      reg [47:0]C_tb;
5      reg clk_tb,rst_n_tb;
6      wire [47:0]P_dut;
7      Simple_DSP #(OPERATION_tb) DUT(A_tb,B_tb,C_tb,D_tb,clk_tb,rst_n_tb,P_dut);
8      initial begin
9          clk_tb=0;
10         forever
11             #1 clk_tb=~clk_tb;
12     end
13     initial begin
14         rst_n_tb=0;
15         D_tb=0;B_tb=0;A_tb=0;C_tb=0;
16         @(negedge clk_tb );
17
18         rst_n_tb=1;D_tb=5;B_tb=5;A_tb=10;C_tb=10;//p=105
19         @(negedge clk_tb );
20
21         D_tb=5;B_tb=5;A_tb=5;C_tb=5;//p=55
22         @(negedge clk_tb );
23
24         D_tb=15;B_tb=15;A_tb=5;C_tb=5;//p=155
25         @(negedge clk_tb );
26
27         D_tb=20;B_tb=20;A_tb=5;C_tb=5;//p=205
28         @(negedge clk_tb );
29
30         D_tb=2;B_tb=5;A_tb=5;C_tb=5;//p=40
31         @(negedge clk_tb );
32
33         D_tb=3;B_tb=5;A_tb=5;C_tb=5;//p=45
34         @(negedge clk_tb );
35
36         D_tb=4;B_tb=5;A_tb=5;C_tb=5;//p=50
37         @(negedge clk_tb );
38
39         D_tb=15;B_tb=7;A_tb=2;C_tb=5;//p=49
40         @(negedge clk_tb );
41
42
43         D_tb=18;B_tb=2;A_tb=4;C_tb=5;//p=85
44         @(negedge clk_tb );
45
46         D_tb=8;B_tb=2;A_tb=9;C_tb=6;//p=96
47         @(negedge clk_tb );
48         D_tb=16;B_tb=1;A_tb=4;C_tb=7;//p=75
49         repeat (5)
50             @(negedge clk_tb );
51         $stop;
52     end
53 endmodule
```

## The do file code:

```
1  vlib work
2
3  vlog DSP.v D_Flipflop.v DSP_BLOCK_tb.v DSP_block_tb.v
4
5  vsim -voptargs=+acc DSP_block_tb
6
7  add wave *
8
9  run -all
10
11 #quit -sim
```

## The constraint code:

```
## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]

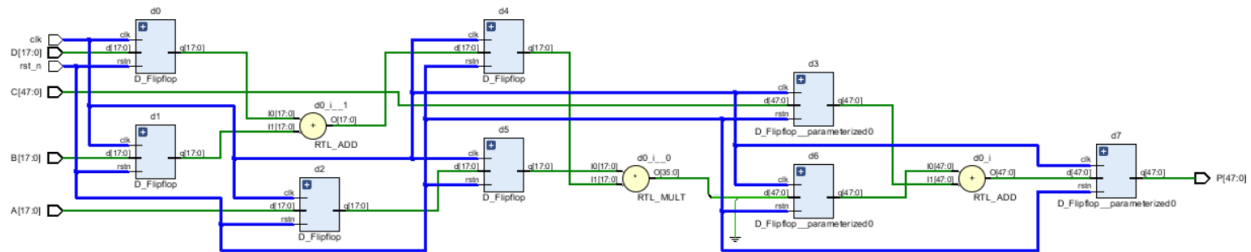
create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
set_property port_width 18 [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets [list {B_IBUF[0]} {B_IBUF[1]} {B_IBUF[2]} {B_IBUF[3]} {B_IBUF[4]} {B_IBUF[5]} {B_IBUF[6]} {B_IBUF[7]} {B_IBUF[8]} {B_IBUF[9]} {B_IBUF[10]} {B_IBUF[11]} {B_IBUF[12]} {B_IBUF[13]} {B_IBUF[14]} {B_IBUF[15]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
set_property port_width 48 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list {P_OBUF[0]} {P_OBUF[1]} {P_OBUF[2]} {P_OBUF[3]} {P_OBUF[4]} {P_OBUF[5]} {P_OBUF[6]} {P_OBUF[7]} {P_OBUF[8]} {P_OBUF[9]} {P_OBUF[10]} {P_OBUF[11]} {P_OBUF[12]} {P_OBUF[13]} {P_OBUF[14]} {P_OBUF[15]} {P_OBUF[16]} {P_OBUF[17]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
set_property port_width 18 [get_debug_ports u_ila_0/probe2]
connect_debug_port u_ila_0/probe2 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]} {A_IBUF[3]} {A_IBUF[4]} {A_IBUF[5]} {A_IBUF[6]} {A_IBUF[7]} {A_IBUF[8]} {A_IBUF[9]} {A_IBUF[10]} {A_IBUF[11]} {A_IBUF[12]} {A_IBUF[13]} {A_IBUF[14]} {A_IBUF[15]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
set_property port_width 48 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list {C_IBUF[0]} {C_IBUF[1]} {C_IBUF[2]} {C_IBUF[3]} {C_IBUF[4]} {C_IBUF[5]} {C_IBUF[6]} {C_IBUF[7]} {C_IBUF[8]} {C_IBUF[9]} {C_IBUF[10]} {C_IBUF[11]} {C_IBUF[12]} {C_IBUF[13]} {C_IBUF[14]} {C_IBUF[15]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
set_property port_width 18 [get_debug_ports u_ila_0/probe4]
connect_debug_port u_ila_0/probe4 [get_nets [list {D_IBUF[0]} {D_IBUF[1]} {D_IBUF[2]} {D_IBUF[3]} {D_IBUF[4]} {D_IBUF[5]} {D_IBUF[6]} {D_IBUF[7]} {D_IBUF[8]} {D_IBUF[9]} {D_IBUF[10]} {D_IBUF[11]} {D_IBUF[12]} {D_IBUF[13]} {D_IBUF[14]} {D_IBUF[15]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
set_property port_width 1 [get_debug_ports u_ila_0/probe5]
connect_debug_port u_ila_0/probe5 [get_nets [list clk_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
set_property port_width 1 [get_debug_ports u_ila_0/probe6]
connect_debug_port u_ila_0/probe6 [get_nets [list rst_n_IBUF]]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
```

## 1-Elaboration (messages\_tab)

The screenshot shows the Vivado Messages window with the following structure:

- Tcl Console** | **Messages** | Log | Reports | Design Runs
- Search, Filter, Sort, and other icons.
- ☒ Info (13) | ☒ Status (11) | Hide All
- ▼ **Vivado Commands** (3 infos, 4 status messages)
  - ▼ **General Messages** (3 infos, 4 status messages)
    - [IP\_Flow 19-234] Refreshing IP repositories
    - [IP\_Flow 19-1704] No user IP repositories specified
    - [IP\_Flow 19-2313] Loaded Vivado IP repository 'D:/Vivado/Vivado/2018.2/data/ip'.
    - > [Command: synth\_design -rtl -name rtl\_1 (3 more like this)]
- ▼ **Elaborated Design** (10 infos, 7 status messages)
  - ▼ **General Messages** (10 infos, 7 status messages)
    - > [Synth 8-6157] synthesizing module 'Simple\_DSP' [DSP.v:1] (2 more like this)

## 2-Elaboration(Schematic)



### 3-synthesis(messages\_tab)

## 4-synthesis(Utilization\_report)

Tcl Console

Messages

Log

Reports

Design Runs

Utilization

Debug

Q

≡

⌵

Hierarchy

Hierarchy

Summary

▼ Slice Logic

▼ Slice LUTs (<1%)

LUT as Logic (<1%)

▼ Slice Registers (<1%)

Register as Flip Flop (<1

Memory

▼ DSP

▼ DSPs (<1%)

DSP48E1 only

▼ IO and GT Specific

Q

≡

⌵

%

Hierarchy

Name	Slice LUTs (134600)	Slice Registers (269200)	DSP s (740 )	Bonded IOB (500)	BUFGCTRL (32)
▼ Simple_DSP	55	151	1	152	1
d0 (D_Flipflop)	18	18	0	0	0
d1 (D_Flipflop_0)	0	18	0	0	0
d2 (D_Flipflop_1)	0	18	0	0	0
d3 (D_Flipflop__para...	0	48	0	0	0
d4 (D_Flipflop_2)	0	1	0	0	0
d6 (D_Flipflop__para...	36	0	1	0	0
d7 (D_Flipflop__para...	1	48	0	0	0

## 5-synthesis(Timing\_report)

Tcl ConsoleMessagesLogReportsDesign RunsTimingUtilizationDebug?

Q

≡

⚙

↺

📁

●

Design Timing Summary

General Information

Timer Settings

Design Timing Summary

Clock Summary (1)

Check Timing (151)

Intra-Clock Paths

Inter-Clock Paths

Other Path Groups

User Ignored Paths

Unconstrained Paths

Setup

Hold

Pulse Width

Worst Negative Slack (WNS): 7.083 ns

Worst Hold Slack (WHS): 0.172 ns

Worst Pulse Width Slack (WPWS): 4.500 ns

Total Negative Slack (TNS): 0.000 ns

Total Hold Slack (THS): 0.000 ns

Total Pulse Width Negative Slack (TPWS): 0.000 ns

Number of Failing Endpoints: 0

Number of Failing Endpoints: 0

Number of Failing Endpoints: 0

Total Number of Endpoints: 86

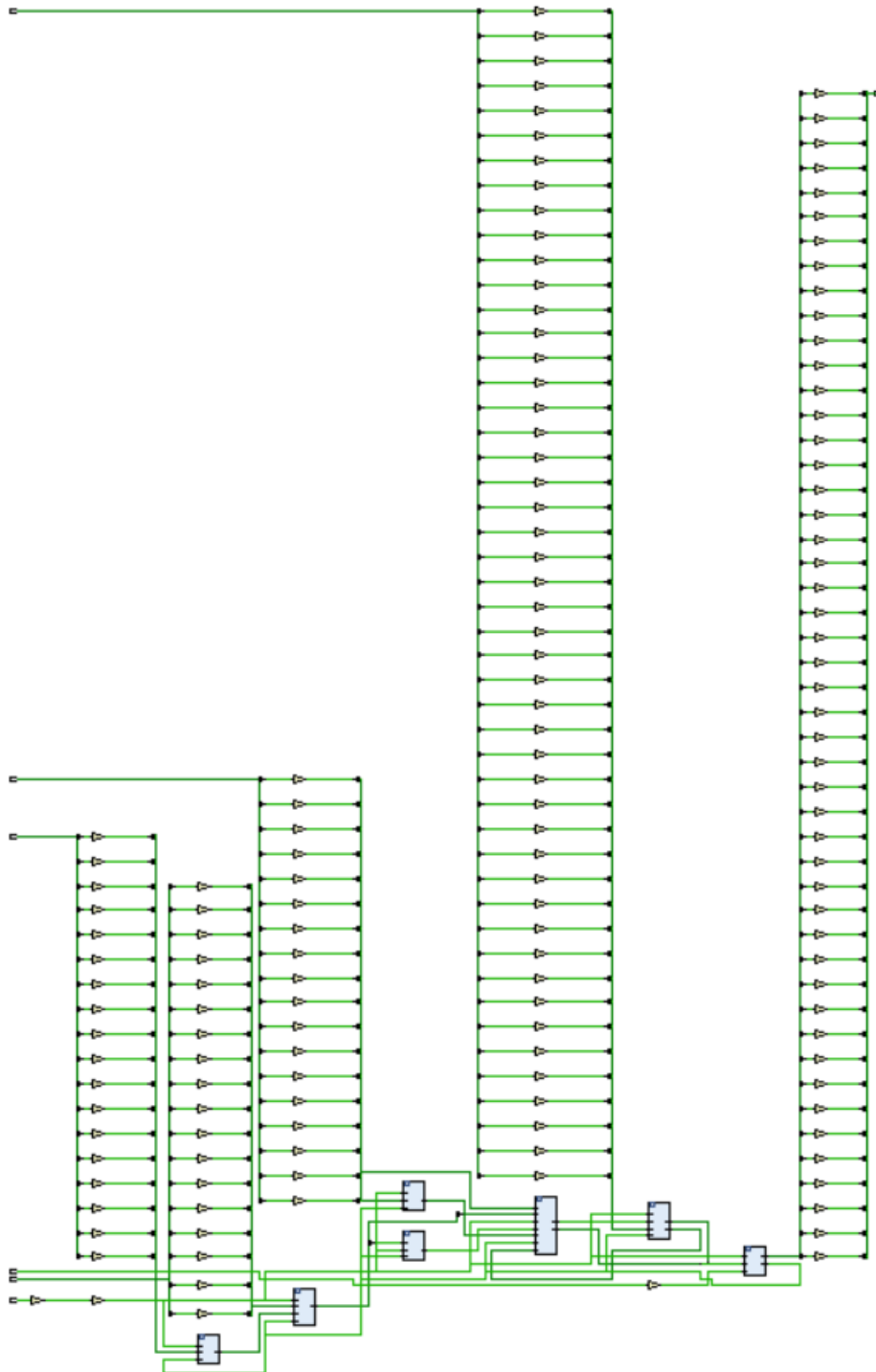
Total Number of Endpoints: 86

Total Number of Endpoints: 153

All user specified timing constraints are met.



## 6-synthesis(Schematic)



## 7-implmentation(messages\_tab)

The Messages tab displays the following warnings:

- Synthesis (18 warnings)**
- Synthesized Design (1 warning)**
- Implementation (2 warnings)**
  - Route Design (2 warnings)**
    - DRC (1 warning)**
      - Pin Planning (1 warning)**
        - [DRC CFGBSV-7] CONFIG\_VOLTAGE with Config Bank VCC0: The CONFIG\_MODE property of current\_design specifies a configuration mode (SPIx4) that uses pins in bank 14. I/O standards used in this bank have a voltage requirement of 1.80. However, the CONFIG\_VOLTAGE for current\_design is set to 3.3. Ensure that your configuration voltage is compatible with the I/O standards in banks used by your configuration mode. Refer to device configuration user guide for more information. Pins used by config mode: V28 (IO\_L1P\_T0\_D00\_M0S\_14), V29 (IO\_L1N\_T0\_D01\_OIN\_14), V26 (IO\_L2P\_T0\_D02\_14), V27 (IO\_L2N\_T0\_D03\_14), W26 (IO\_L3P\_T0\_D05\_PUDC\_B\_14), and Y27 (IO\_L6P\_T0\_FCS\_B\_14)
      - [Timing 38-436] There are set\_bus\_skew constraint(s) in this design. Please run report\_bus\_skew to ensure that bus skew requirements are met.
  - Implemented Design (2 warnings)**
    - General Messages (2 warnings)**
      - [Timing 38-436] There are set\_bus\_skew constraint(s) in this design. Please run report\_bus\_skew to ensure that bus skew requirements are met. (1 more like this)

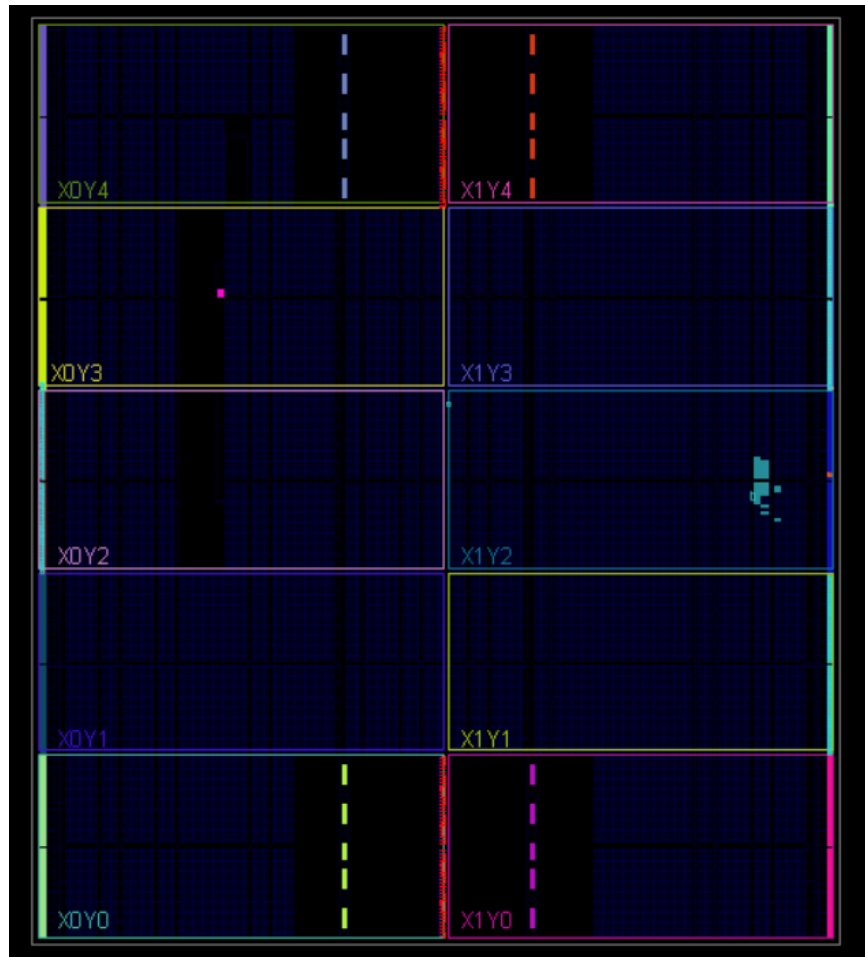
## 8-Implmentation(Utilization\_report)

Reports	Design Runs	DRC	Methodology	Power	Timing	Utilization					
Hierarchy											
Name	Slice LUTs (133800)	Slice Registers (267600)	F7 Muxes (66900)	Slice (3345 0)	LUT as Logic (133800)	LUT as Memory (46200)	LUT Flip Flop Pairs (133800)	Block RAM Tile (365)	DSP s (740 )	Bonded IOB (500)	
Simple_DSP1	1575	2786	12	706	1321	254	961	4.5	1	152	
d0 (D_Flipflop)	18	18	0	13	18	0	0	0	0	0	
d1 (D_Flipflop_0)	0	18	0	7	0	0	0	0	0	0	
d2 (D_Flipflop_1)	0	18	0	5	0	0	0	0	0	0	
d3 (D_Flipflop__para...	0	48	0	16	0	0	0	0	0	0	
d4 (D_Flipflop_2)	0	1	0	1	0	0	0	0	0	0	
d6 (D_Flipflop__para...	36	0	0	9	36	0	0	0	1	0	
d7 (D_Flipflop__para...	1	48	0	13	1	0	0	0	0	0	
dbg_hub (dbg_hub)	474	727	0	222	450	24	316	0	0	0	
u_ila_0 (u_ila_0)	1046	1908	12	464	816	230	605	4.5	0	0	

## 9-Implmentation(Timing\_report)

Reports	Design Runs	DRC	Methodology	Power	Timing	Utilization
Design Timing Summary						
Setup		Hold		Pulse Width		
Worst Negative Slack (WNS): 3.344 ns		Worst Hold Slack (WHS): 0.057 ns		Worst Pulse Width Slack (WPWS): 3.950 ns		
Total Negative Slack (TNS): 0.000 ns		Total Hold Slack (THS): 0.000 ns		Total Pulse Width Negative Slack (TPWS): 0.000 ns		
Number of Failing Endpoints: 0		Number of Failing Endpoints: 0		Number of Failing Endpoints: 0		
Total Number of Endpoints: 5267		Total Number of Endpoints: 5251		Total Number of Endpoints: 3273		
All user specified timing constraints are met.						

## 10-Implmentation(device)



snippets of wave:

